Pattern Clustering Using a Swarm Intelligence Approach

Swagatam Das[1] and and Ajith Abraham[2]

[1] Department of Electronics and Telecommunication Engineering,
Jadavpur University, Kolkata 700032, India.
[2] Center of Excellence for Quantifiable Quality of Service
Norwegian University of Science and Technology,
Trondheim, Norway
ajith.abraham@ieee.org

Summary. Clustering aims at representing large datasets by a fewer number of prototypes or clusters. It brings simplicity in modeling data and thus plays a central role in the process of knowledge discovery and data mining. Data mining tasks, in these days, require fast and accurate partitioning of huge datasets, which may come with a variety of attributes or features. This, in turn, imposes severe computational requirements on the relevant clustering techniques. A family of bio-inspired algorithms, well-known as Swarm Intelligence (SI) has recently emerged that meets these requirements and has successfully been applied to a number of real world clustering problems. This chapter explores the role of SI in clustering different kinds of datasets. It finally describes a new SI technique for partitioning a linearly non-separable dataset into an optimal number of clusters in the kernel- induced feature space. Computer simulations undertaken in this research have also been provided to demonstrate the effectiveness of the proposed algorithm.

1 Introduction

Clustering means the act of partitioning an unlabeled dataset into groups of similar objects. Each group, called a 'cluster', consists of objects that are similar between themselves and dissimilar to objects of other groups. In the past few decades, cluster analysis has played a central role in a variety of fields ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer sciences (web mining, spatial database analysis, textual document collection, image segmentation), life and medical sciences (genetics, biology, microbiology, paleontology, psychiatry, pathology), to earth sciences (geography. geology, remote sensing), social sciences (sociology, psychology, archeology, education), and economics (marketing, business) (Evangelou *et al*., 2001, Lillesand and Keifer, 1994, Rao, 1971, Duda and Hart, 1973, Everitt, 1993, Xu and Wunsch, 2008).

Human beings possess the natural ability of clustering objects. Given a box full of marbles of four different colors say red, green, blue, and yellow, even a child may separate these marbles into four clusters based on their colors. However, making a computer solve this type of problems is quite difficult and demands the attention of computer scientists and engineers all

over the world till date. The major hurdle in this task is that the functioning of the brain is much less understood. The mechanisms, with which it stores huge amounts of information, processes them at lightning speeds and infers meaningful rules, and retrieves information as and when necessary have till now eluded the scientists. A question that naturally comes up is: what is the point in making a computer perform clustering when people can do this so easily? The answer is far from trivial. The most important characteristic of this information age is the abundance of data. Advances in computer technology, in particular the Internet, have led to what some people call "data explosion": the amount of data available to any person has increased so much that it is more than he or she can handle. In reality the amount of data is vast and in addition, each data item (an abstraction of a real-life object) may be characterized by a large number of attributes (or *features*), which are based on certain measurements taken on the real-life objects and may be numerical or non-numerical. Mathematically we may think of a mapping of each data item into a point in the multi-dimensional feature space (each dimension corresponding to one feature) that is beyond our perception when number of features exceed just 3. Thus it is nearly impossible for human beings to partition tens of thousands of data items, each coming with several features (usually much greater than 3), into meaningful clusters within a short interval of time. Nonetheless, the task is of paramount importance for organizing and summarizing huge piles of data and discovering useful knowledge from them. So, can we devise some means to generalize to arbitrary dimensions of what humans perceive in two or three dimensions, as densely connected "patches" or "clouds" within data space? The entire research on cluster analysis may be considered as an effort to find satisfactory answers to this fundamental question.

The task of computerized data clustering has been approached from diverse domains of knowledge like graph theory, statistics (multivariate analysis), artificial neural networks, fuzzy set theory, and so on (Forgy, 1965, Zahn, 1971, Mitchell, 1997, Mao and Jain, 1995, Pal *et al*., 1993, Kohonen, 1995, Falkenauer, 1998, Paterlini and Minerva, 2003, Xu and Wunsch, 2005,Rokach and Maimon, 2005,Mitra *et al*.2002). One of the most popular approaches in this direction has been the formulation of clustering as an optimization problem, where the best partitioning of a given dataset is achieved by minimizing/maximizing one (single-objective clustering) or more (multi-objective clustering) objective functions. The objective functions are usually formed capturing certain statistical-mathematical relationship among the individual data items and the candidate set of representatives of each cluster (also known as cluster-centroids). The clusters are either hard, that is each sample point is unequivocally assigned to a cluster and is considered to bear no similarity to members of other clusters, or fuzzy, in which case a membership function expresses the degree of belongingness of a data item to each cluster.

Most of the classical optimization-based clustering algorithms (including the celebrated hard c-means and fuzzy c-means algorithms) rely on local search techniques (like iterative function optimization, Lagrange's multiplier, Picard's iterations etc.) for optimizing the clustering criterion functions. The local search methods, however, suffer from two great disadvantages. Firstly they are prone to getting trapped in some local optima of the multi-dimensional and usually multi-modal landscape of the objective function. Secondly performances of these methods are usually very sensitive to the initial values of the search variables.

Although many respected texts of pattern recognition describe clustering as an unsupervised learning method, most of the traditional clustering algorithms require a prior specification of the number of clusters in the data for guiding the partitioning process, thus making it not completely unsupervised. On the other hand, in many practical situations, it is impossible to provide even an estimation of the number of naturally occurring clusters in a previously unhandled dataset. For example, while attempting to classify a large database of handwritten

characters in an unknown language; it is not possible to determine the correct number of distinct letters beforehand. Again, while clustering a set of documents arising from the query to a search engine, the number of classes can change for each set of documents that result from an interaction with the search engine. Data mining tools that predict future trends and behaviors for allowing businesses to make proactive and knowledge-driven decisions, demand fast and fully automatic clustering of very large datasets with minimal or no user intervention. Thus it is evident that the complexity of the data analysis tasks in recent times has posed severe challenges before the classical clustering techniques.

Recently a family of nature inspired algorithms, known as *Swarm Intelligence* (SI), has attracted several researchers from the field of pattern recognition and clustering. Clustering techniques based on the SI tools have reportedly outperformed many classical methods of partitioning a complex real world dataset. Algorithms belonging to the domain, draw inspiration from the collective intelligence emerging from the behavior of a group of social insects (like bees, termites and wasps). When acting as a community, these insects even with very limited individual capability can jointly (cooperatively) perform many complex tasks necessary for their survival. Problems like finding and storing foods, selecting and picking up materials for future usage require a detailed planning, and are solved by insect colonies without any kind of supervisor or controller. An example of particularly successful research direction in swarm intelligence is Ant Colony Optimization (ACO) (Dorigo *et al*., 1996,Dorigo and Gambardella, 1997), which focuses on discrete optimization problems, and has been applied successfully to a large number of NP hard discrete optimization problems including the traveling salesman, the quadratic assignment, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks. Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) is another very popular SI algorithm for global optimization over continuous search spaces. Since its advent in 1995, PSO has attracted the attention of several researchers all over the world resulting into a huge number of variants of the basic algorithm as well as many parameter automation strategies.

In this Chapter, we explore the applicability of these bio-inspired approaches to the development of self-organizing, evolving, adaptive and autonomous clustering techniques, which will meet the requirements of next-generation data mining systems, such as diversity, scalability, robustness, and resilience. The next section of the chapter provides an overview of the SI paradigm with a special emphasis on two SI algorithms well-known as Particle Swarm Optimization (PSO) and Ant Colony Systems (ACS). Section 3 outlines the data clustering problem and briefly reviews the present state of the art in this field. Section 4 describes the use of the SI algorithms in both crisp and fuzzy clustering of real world datasets. A new automatic clustering algorithm, based on PSO, is presented in Section 5. The algorithm requires no previous knowledge of the dataset to be partitioned, and can determine the optimal number of classes dynamically in a linearly non-separable dataset using a kernel-induced distance metric. The new method has been compared with two well-known, classical fuzzy clustering algorithms. The Chapter is concluded in Section 6 with discussions on possible directions for future research.

2 An Introduction to Swarm Intelligence

The behavior of a single ant, bee, termite and wasp often is too simple, but their collective and social behavior is of paramount significance. A look at National Geographic TV Channel reveals that advanced mammals including lions also enjoy social lives, perhaps for their self-existence at old age and in particular when they are wounded. The collective and social

behavior of living creatures motivated researchers to undertake the study of today what is known as *Swarm Intelligence*. Historically, the phrase Swarm Intelligence (SI) was coined by Beny and Wang in late 1980s (Beni and Wang, 1989) in the context of cellular robotics. A group of researchers in different parts of the world started working almost at the same time to study the versatile behavior of different living creatures and especially the social insects. The efforts to mimic such behaviors through computer simulation finally resulted into the fascinating field of SI. SI systems are typically made up of a population of simple agents (an entity capable of performing/executing certain operations) interacting locally with one another and with their environment. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior. Many biological creatures such as fish schools and bird flocks clearly display structural order, with the behavior of the organisms so integrated that even though they may change shape and direction, they appear to move as a single coherent entity (Couzin *et al*., 2002). The main properties of the collective behavior can be pointed out as follows and is summarized in Figure 1.

1. ***Homogeneity*:** every bird in flock has the same behavioral model. The flock moves without a leader, even though temporary leaders seem to appear.
2. ***Locality*:** its nearest flock-mates only influence the motion of each bird. Vision is considered to be the most important senses for flock organization.
3. ***Collision Avoidance*:** avoid colliding with nearby flock mates.
4. ***Velocity Matching:*** attempt to match velocity with nearby flock mates.
5. ***Flock Centering*:** attempt to stay close to nearby flock mates

Individuals attempt to maintain a minimum distance between themselves and others at all times. This rule is given the highest priority and corresponds to a frequently observed behavior of animals in nature (Krause and Ruxton, 2002). If individuals are not performing an avoidance maneuver they tend to be attracted towards other individuals (to avoid being isolated) and to align themselves with neighbors (Partridge and Pitcher, 1980, Partridge, 1982).

Couzin *et al.* (2002) identified four collective dynamical behaviors as illustrated in Figure 2:

1. ***Swarm:*** an aggregate with cohesion, but a low level of polarization (parallel alignment) among members
2. ***Torus*:** individuals perpetually rotate around an empty core (milling). The direction of rotation is random.
3. ***Dynamic parallel group*:** the individuals are polarized and move as a coherent group, but individuals can move throughout the group and density and group form can fluctuate (Partridge and Pitcher, 1980, Major and Dill, 1978).
4. ***Highly parallel group*:** much more static in terms of exchange of spatial positions within the group than the dynamic parallel group and the variation in density and form is minimal.

As mentioned in (Grosan *et al*., 2006) at a high-level, a swarm can be viewed as a group of agents cooperating to achieve some purposeful behavior and achieve some goal (Abraham *et al*., 2006). This collective intelligence seems to emerge from what are often large groups: According to Milonas (1994), five basic principles define the SI paradigm. First is the proximity principle: the swarm should be able to carry out simple space and time computations. Second is the quality principle: the swarm should be able to respond to quality factors in the environment. Third is the principle of diverse response: the swarm should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the swarm should

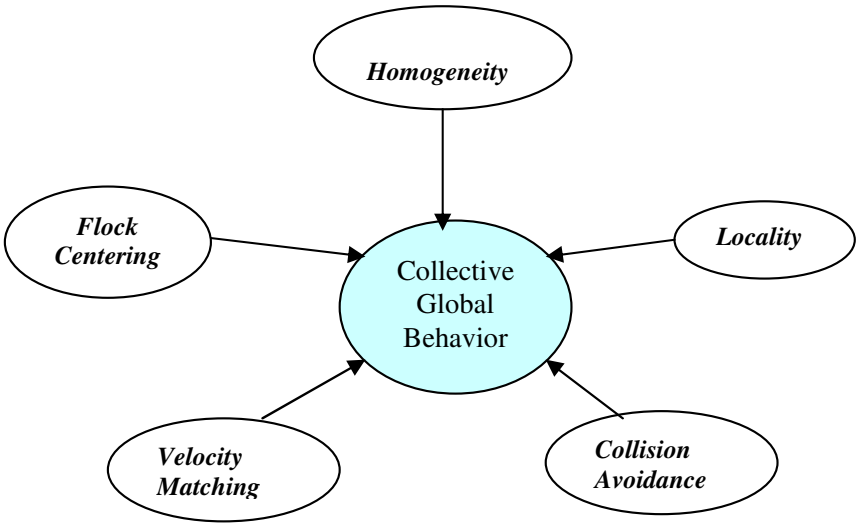


Fig. 1. Main traits of collective behavior.

not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the swarm must be able to change behavior mote when it is worth the computational price. Note that principles four and five are the opposite sides of the same coin. Below we discuss in details two algorithms from SI domain, which have gained wide popularity in a relatively short span of time.

2.1 The Ant Colony Systems

The basic idea of a real ant system is illustrated in Figure 3. In the left picture, the ants move in a straight line to the food. The middle picture illustrates the situation soon after an obstacle is inserted between the nest and the food. To avoid the obstacle, initially each ant chooses to turn left or right at random. Let us assume that ants move at the same speed depositing pheromone in the trail uniformly. However, the ants that, by chance, choose to turn left will reach the food sooner, whereas the ants that go around the obstacle turning right will follow a longer path, and so will take longer time to circumvent the obstacle. As a result, pheromone accumulates faster in the shorter path around the obstacle. Since ants prefer to follow trails with larger amounts of pheromone, eventually all the ants converge to the shorter path around the obstacle, as shown in Figure 3.

An artificial Ant Colony System (ACS) is an agent-based system, which simulates the natural behavior of ants and develops mechanisms of cooperation and learning. ACS was proposed by Dorigo *et al.* (1997) as a new heuristic to solve combinatorial optimization problems. This new heuristic, called Ant Colony Optimization (ACO) has been found to be both robust and versatile in handling a wide range of combinatorial optimization problems.

The main idea of ACO is to model a problem as the search for a minimum cost path in a graph. Artificial ants as if walk on this graph, looking for cheaper paths. Each ant has a rather

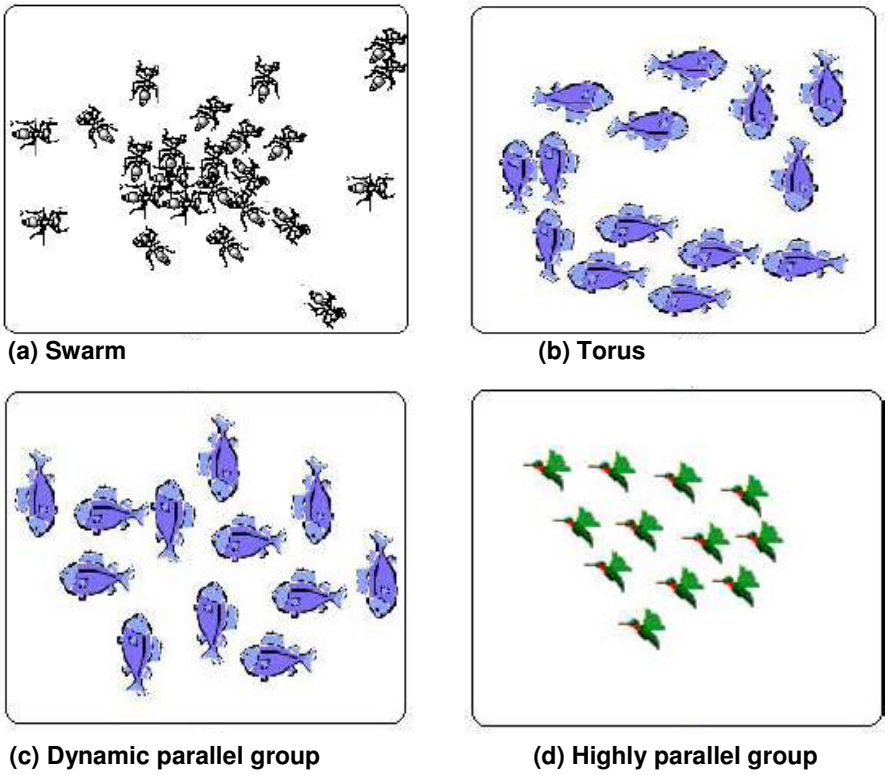


Fig. 2. Different models of collective behavior.

simple behavior capable of finding relatively costlier paths. Cheaper paths are found as the emergent result of the global cooperation among ants in the colony. The behavior of artificial ants is inspired from real ants: they lay pheromone trails (obviously in a mathematical form) on the graph edges and choose their path with respect to probabilities that depend on pheromone trails. These pheromone trails progressively decrease by evaporation. In addition, artificial ants have some extra features not seen in their counterpart in real ants. In particular, they live in a discrete world (a graph) and their moves consist of transitions from nodes to nodes.

Below we illustrate the use of ACO in finding the optimal tour in the classical Traveling Salesman Problem (TSP). Given a set of n cities and a set of distances between them, the problem is to determine a minimum traversal of the cities and return to the home-station at the end. It is indeed important to note that the traversal should in no way include a city more than once. Let $r\ (Cx, Cy)$ be a measure of cost for traversal from city Cx to Cy. Naturally, the total cost of traversing n cities indexed by $i1, i2, i3, \ldots, in$ in order is given by the following expression:

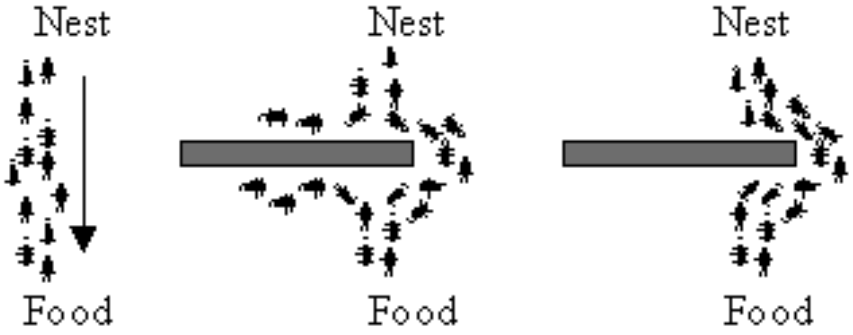


Fig. 3. Illustrating the behavior of real ant movements.

$$Cost(i_1, i_2, ..., i_n) = \sum_{j=1}^{n-1} r(Ci_j, Ci_{j+1}) + r(Ci_n, Ci_1) \tag{1}$$

The ACO algorithm is employed to find an optimal order of traversal of the cities. Let τ be a mathematical entity modeling the pheromone and $\eta ij = 1/r$ (i, j) is a local heuristic. Also let *allowedk(t)* be the set of cities that are yet to be visited by ant q located in city i. Then according to the classical ant system (Xu and Wunsch, 2008) the probability that ant q in city i visits city j is given by

$$p_{ij}^{q}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{h \in allowed_q(t)} \left[\tau_{ih}(t)\right]^{\alpha} \cdot \left[\eta_{ih}\right]^{\beta}}, \quad \text{if j} \in allowed_k(t)$$

$$= \quad 0, \qquad \text{otherwise.} \tag{2}$$

In Equation 19 shorter edges with greater amount of pheromone are favored by multiplying the pheromone on edge (i, j) by the corresponding heuristic value $\eta(i, j)$. Parameters α (> 0) and β (> 0) determine the relative importance of pheromone versus cost. Now in ant system, pheromone trails are updated as follows. Let Dq be the length of the tour performed by ant q, $\Delta\tau_q(i, j) = 1/D_q$ if $(i, j) \in$ tour done by ant q and $\Delta\tau_q(i, j) = 0$ otherwise and finally let $\rho \in [0,1]$ be a pheromone decay parameter which takes care of the occasional evaporation of the pheromone from the visited edges. Then once all ants have built their tours, pheromone is updated on all the ages as,

$$\tau(i, j) = (1 - \rho).\tau(i, j) + \sum_{p=1}^{m} \tau_k(i, j) \tag{3}$$

From equation 3, we can guess that pheromone updating attempts to accumulate greater amount of pheromone to shorter tours (which corresponds to high value of the second term in (3) so as to compensate for any loss of pheromone due to the first term). This conceptually resembles a reinforcement-learning scheme, where better solutions receive a higher reinforcement.

The ACO differs from the classical ant system in the sense that here the pheromone trails are updated in two ways. Firstly, when ants construct a tour they locally change the amount of pheromone on the visited edges by a local updating rule. Now if we let γ to be a decay parameter and $\Delta\tau(i,j) = \tau 0$ such that $\tau 0$ is the initial pheromone level, then the local rule may be stated as:

$$\tau(i,j) = (1-\gamma).\tau(i,j) + \gamma.\Delta\tau(i,j) \tag{4}$$

Secondly, after all the ants have built their individual tours, a global updating rule is applied to modify the pheromone level on the edges that belong to the best ant tour found so far. If κ be the usual pheromone evaporation constant, D_{gb} be the length of the globally best tour from the beginning of the trial and $\Delta\tau'/(i\ ,\ j) = 1/\ D_{gb}$ only when the edge ($i,\ j$) belongs to global-best-tour and zero otherwise, then we may express the global rule as follows:

$$\tau(i,j) = (1-\kappa).\tau(i,j) + \kappa.\Delta\tau'(i,j) \tag{5}$$

The main steps of ACO algorithm are presented below.

Procedure ACO
Begin
Initialize pheromone trails;
Repeat
Begin /* at this stage each loop is called an iteration */
Each ant is positioned on a starting node;
Repeat
Begin /* at this level each loop is called a step */
Each ant applies a *state transition rule like rule (2)* to incrementally build a solution and a *local pheromone-updating rule like rule (4);*
Until all ants have built a complete solution;
A *global pheromone-updating rule like rule (5)* is applied.
Until terminating condition is reached;
End

The concept of Particle Swarms, although initially introduced for simulating human social behaviors, has become very popular these days as an efficient search and optimization technique. The Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995, Kennedy *et al*., 2001), as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators and is conceptually very simple. In PSO, a population of conceptual 'particles' is initialized with random positions $\mathbf{X}_i$ and velocities $\mathbf{V}_i$, and a function, f, is evaluated, using the particle's positional coordinates as input values. In an D-dimensional search space, $\mathbf{X}_i = (x_{i1}, x_{i2}, ..., x_{iD})^T$ and $\mathbf{V}_i = (v_{i1}, v_{i2}, ..., v_{iD})^T$. In literature, the basic equations for updating the d-th dimension of the velocity and position of the i-th particle for PSO are presented most popularly in the following way:

$$\begin{aligned} v_{i,d}(t) = \omega.v_{i,d}(t-1) + \varphi_1.rand1_{i,d}(0,1).(p^l_{i,d} - x_{i,d}(t-1)) + \\ \varphi_2.rand2_{i,d}(0,1).(p^g_d - x_{i,d}(t-1)) \end{aligned} \tag{6}$$

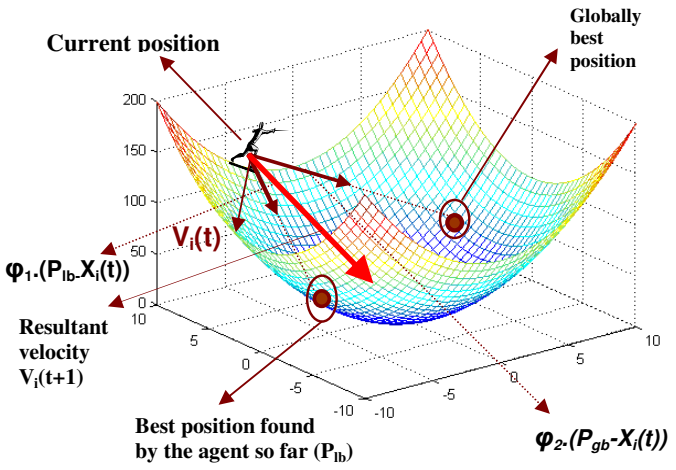


Fig. 4. Illustrating the velocity updating scheme of basic PSO.

$$x_{i,d}(t) = x_{i,d}(t-1) + v_{i,d}(t) \tag{7}$$

Please note that in 6 and 10, φ_1 and φ_2 are two positive numbers known as the *acceleration coefficients*. The positive constant $\omega$is known as *inertia factor*. $rand1_{i,d}(0,1)$ and $rand2_{i,d}(0,1)$ are the two uniformly distributed random numbers in the range of [0, 1]. While applying PSO, we define a maximum velocity $\mathbf{V}_{\max} = [v_{\max,1}, v_{\max,2}, ..., v_{\max,D}]^T$ of the particles in order to control their convergence behavior near optima. If $|v_{i,d}|$ exceeds a positive constant value $v_{\max,d}$ specified by the user, then the velocity of that dimension is assigned to $sgn(v_{i,d}).v_{\max,d}$ where *sgn* stands for the signum function and is defined as:

$$\begin{aligned} \text{sgn}(x) &= 1, && \text{if } x > 0 \\ &= 0, && \text{if } x = 0 \\ &= -1, && \text{if } x < 0 \end{aligned} \tag{8}$$

While updating the velocity of a particle, different dimensions will have different values for *rand*1 and *rand*2. Some researchers, however, prefer to use the same values of these random coefficients for all dimensions of a given particle. They use the following formula to update the velocities of the particles:

$$\begin{aligned} v_{i,d}(t) = \omega.v_{i,d}(t-1) + \varphi_1.rand1_i(0,1).(p^l_{i,d}(t) - x_{i,d}(t-1)) + \\ \varphi_2.rand2_i(0,1).(p^g_d(t) - x_{i,d}(t-1)) \end{aligned} \tag{9}$$

Comparing the two variants in 6 and 12, the former can have a larger search space due to independent updating of each dimension, while the second is dimension-dependent and has a smaller search space due to the same random numbers being used for all dimensions The velocity updating scheme has been illustrated in Figure 4 with a humanoid particle.

A pseudo code for the PSO algorithm may be put forward as:

The PSO Algorithm

Input: Randomly initialized position and velocity of the particles: $\vec{X}_i(0)$ and $\vec{V}_i(0)$

Output: Position of the approximate global optima $\vec{X}^*$

Begin
  While terminating condition is not reached **do**
    Begin
      for i = 1 to number of particles
        Evaluate the fitness: = $f(\vec{X}_i(t))$;
        Update $\vec{P}(t)$ and $\vec{g}(t)$;
        Adapt velocity of the particle using equation (6);
        Update the position of the particle;
        increase i;
  end while
end

3 Data Clustering – An Overview

In this Section, we first provide a brief and formal description of the clustering problem. We then discuss a few major classical clustering techniques.

3.1 Problem Definition

A *pattern* is a physical or abstract structure of objects. It is distinguished from others by a collective set of attributes called *features*, which together represent a pattern (Konar, 2005). Let P = {P1, P2... Pn} be a set of n patterns or data points, each having d features. These patterns can also be represented by a profile data matrix **X**n×d having n d-dimensional row vectors. The i-th row vector**X**$_i$characterizes the i-th object from the set P and each element Xi,j in **X**$_i$corresponds to the j-th real value feature (j = 1, 2, ...,d) of the i-th pattern (i =1,2,...., n). Given such an **X**$n \times d$, a partitional clustering algorithm tries to find a partition C = {C1, C2,....., Ck}of k classes, such that the similarity of the patterns in the same cluster is maximum and patterns from different clusters differ as far as possible. The partitions should maintain the following properties:

- Each cluster should have at least one pattern assigned i. e. $C_i \neq \Phi \forall i \in \{1,2,...,k\}$.
- Two different clusters should have no pattern in common. i.e. $C_i \bigcap C_j = \Phi, \forall i \neq j$ and $i, j \in \{1,2,...,k\}$. This property is required for crisp (hard) clustering. In Fuzzy clustering this property doesn't exist.
- Each pattern should definitely be attached to a cluster i.e. $\bigcup_{i=1}^{k} C_i = P$.

Since the given dataset can be partitioned in a number of ways maintaining all of the above properties, a fitness function (some measure of the adequacy of the partitioning) must

be defined. The problem then turns out to be one of finding a partition **C*** of optimal or near-optimal adequacy as compared to all other feasible solutions **C** = { C1, C2,....., CN(n,*k*)} where,

$$N(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^i \binom{k}{i}^i (k-i)^i \tag{10}$$

is the number of feasible partitions. This is same as,

$$Optimize_C f(X_{\mathrm{n}\times\mathrm{d}}, C) \tag{11}$$

where C is a single partition from the set **C** and f is a statistical-mathematical function that quantifies the goodness of a partition on the basis of the similarity measure of the patterns. Defining an appropriate similarity measure plays fundamental role in clustering (Jain *et al*., 1999). The most popular way to evaluate similarity between two patterns amounts to the use of *distance measure*. The most widely used distance measure is the Euclidean distance, which between any two d-dimensional patterns $\mathbf{X}_i$ and $\mathbf{X}_j$ is given by,

$$d(\mathbf{X}_i, \mathbf{X}_j) = \sqrt{\sum_{p=1}^{d}(X_{i,p} - X_{j,p})^2} = \left\|\mathbf{X}_i - \mathbf{X}_j\right\| \tag{12}$$

It has been shown in (Brucker, 1978) that the clustering problem is NP-hard when the number of clusters exceeds 3.

3.2 The Classical Clustering Algorithms

Data clustering is broadly based on two approaches: *hierarchical* and *partitional* (Frigui and Krishnapuram, 1999, Leung *et al*., 2000). Within each of the types, there exists a wealth of subtypes and different algorithms for finding the clusters. In hierarchical clustering, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set (Leung *et al*., 2000). Hierarchical algorithms can be agglomerative (bottom-up) or divisive (top-down). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters. Hierarchical algorithms have two basic advantages (Frigui and Krishnapuram, 1999). Firstly, the number of classes need not be specified a priori and secondly, they are independent of the initial conditions. However, the main drawback of hierarchical clustering techniques is they are static, i.e. data-points assigned to a cluster can not move to another cluster. In addition to that, they may fail to separate overlapping clusters due to lack of information about the global shape or size of the clusters (Jain *et al*., 1999).

Partitional clustering algorithms, on the other hand, attempt to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria. The criterion function may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically, the global criteria involve minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters. The advantages of the hierarchical algorithms are the disadvantages of the partitional algorithms and vice versa. An extensive survey of various clustering techniques can be found in (Jain *et al*., 1999). The focus of this chapter is on the partitional clustering algorithms.

Clustering can also be performed in two different modes: crisp and fuzzy. In crisp clustering, the clusters are disjoint and non-overlapping in nature. Any pattern may belong to one and only one class in this case. In case of fuzzy clustering, a pattern may belong to all the classes with a certain fuzzy membership grade (Jain *et al*., 1999).

The most widely used iterative k-means algorithm (MacQueen, 1967) for partitional clustering aims at minimizing the ICS (Intra-Cluster Spread) which for k cluster centers can be defined as

$$ICS(C_1, C_2, ..., C_k) = \sum_{i=1}^{k} \sum_{\mathbf{X}_i \in C_i} \|\mathbf{X}_i - \mathbf{m}_i\|^2 \tag{13}$$

The k-means (or hard c-means) algorithm starts with k cluster-centroids (these centroids are initially selected randomly or derived from some a priori information). Each pattern in the data set is then assigned to the closest cluster-centre. Centroids are updated by using the mean of the associated patterns. The process is repeated until some stopping criterion is met.

In the c-medoids algorithm (Kaufman and Rousseeuw, 1990), on the other hand, each cluster is represented by one of the representative objects in the cluster located near the center. Partitioning around medoids (PAM) (Kaufman and Rousseeuw, 1990) starts from an initial set of medoids, and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering. Although PAM works effectively for small data, it does not scale well for large datasets. Clustering large applications based on randomized search (CLARANS) (Ng and Han, 1994), using randomized sampling, is capable of dealing with the associated scalability issue.

The fuzzy c-means (FCM) (Bezdek, 1981) seems to be the most popular algorithm in the field of fuzzy clustering. In the classical FCM algorithm, a *within cluster sum* function Jm is minimized to evolve the proper cluster centers:

$$J_m = \sum_{j=1}^{n} \sum_{i=1}^{c} (u_{ij})^m \left\|\mathbf{X}_j - \mathbf{V}_i\right\|^2 \tag{14}$$

where $\mathbf{V}_i$ is the i-th cluster center, $\mathbf{X}_j$ is the j-th d-dimensional data vector and ||. ||is an inner product-induced norm in d dimensions. Given c classes, we can determine their cluster centers $\mathbf{V}_i$ for i=1 to c by means of the following expression:

$$\mathbf{V}_i = \frac{\sum_{j=1}^{n} (u_{ij})^m \mathbf{X}_j}{\sum_{j=1}^{n} (u_{ij})^m} \mathbf{V}_i = \frac{\sum_{j=1}^{n} (u_{ij})^m \mathbf{X}_j}{\sum_{j=1}^{n} (u_{ij})^m} \tag{15}$$

Here m (m>1) is any real number that influences the membership grade. Now differentiating the performance criterion with respect to $\mathbf{V}_i$ (treating uij as constants) and with respect to uij (treating $\mathbf{V}_i$ as constants) and setting them to zero the following relation can be obtained:

$$u_{ij} = \left[\sum_{k=1}^{c} \left(\frac{\left\|\mathbf{X}_j - \mathbf{V}_i\right\|^2}{\|\mathbf{X} - \mathbf{V}_i\|^2} \right)^{1/(m-1)} \right]^{-1} \tag{16}$$

Several modifications of the classical FCM algorithm can be found in (Hall *et al*.1999, Gath and Geva, 1989, Bensaid *et al*., 1996, Clark *et al*., 1994, Ahmed *et al*., 2002, Wang X *et al*., 2004).

3.3 Relevance of SI Algorithms in Clustering

From the discussion of the previous Section, we see that the SI algorithms are mainly stochastic search and optimization techniques, guided by the principles of collective behaviour and self organization of insect swarms. They are efficient, adaptive and robust search methods producing near optimal solutions and have a large amount of implicit parallelism. On the other hand, data clustering may be well formulated as a difficult global optimization problem; thereby making the application of SI tools more obvious and appropriate.

4 Clustering with the SI Algorithms

In this Section we first review the present state of the art clustering algorithms based on SI tools, especially the ACO and PSO. We then outline a new algorithm which employs the PSO model to automatically determine the number of clusters in a previously unhandled dataset. Computer simulations undertaken for this study have also been included to demonstrate the elegance of the new dynamic clustering technique.

4.1 The Ant Colony Based Clustering Algorithms

Ant colonies provide a means to formulate some powerful nature-inspired heuristics for solving the clustering problems. Among other social movements, researchers have simulated the way, ants work collaboratively in the task of grouping dead bodies so, as to keep the nest clean (Bonabeau *et al*., 1999). It can be observed that, with time the ants tend to cluster all dead bodies in a specific region of the environment, thus forming piles of corpses.

Larval sorting and corpse cleaning by ant was first modeled by Deneubourg *et al.* (1991) for accomplishing certain tasks in robotics. This inspired the Ant-based clustering algorithm (Handl *et al*., 2003). Lumer and Faieta modified the algorithm using a dissimilarity-based evaluation of the local density, in order to make it suitable for data clustering (Lumer and Faieta, 1994). This introduced standard Ant Clustering Algorithm (ACA). It has subsequently been used for numerical data analysis (Lumer and Faieta, 1994), data-mining (Lumer and Faieta, 1995), graph-partitioning (Kuntz and Snyers, 1994, Kuntz and Snyers, 1999, Kuntz *et al*., 1998) and text-mining (Handl and Meyer B, 2002, Hoe *et al*., 2002, Ramos and Merelo, 2002). Many authors (Handl and Meyer B, 2002, Ramos *et al*., 2002) proposed a number of modifications to improve the convergence rate and to get optimal number of clusters. Monmarche *et al.* (1999) hybridized the Ant-based clustering algorithm with k-means algorithm and compared it to traditional k-means on various data sets, using the classification error for evaluation purposes. However, the results obtained with this method are not applicable to ordinary ant-based clustering since it differs significantly from the latter.

Like a standard ACO, ant-based clustering is a distributed process that employs positive feedback. Ants are modeled by simple agents that randomly move in their environment. The environment is considered to be a low dimensional space, more generally a two-dimensional plane with square grid. Initially, each data object that represents a multi-dimensional pattern is randomly distributed over the 2-D space. Data items that are scattered within this environment can be picked up, transported and dropped by the agents in a probabilistic way. The picking and dropping operation are influenced by the similarity and density of the data items within the ant's local neighborhood. Generally, the size of the neighborhood is 3×3. Probability of picking up data items is more when the object are either isolated or surrounded by dissimilar

items. They trend to drop them in the vicinity of similar ones. In this way, a clustering of the elements on the grid is obtained.

The ants search for the feature space either through random walk or with jumping using a short term memory. Each ant picks up or drops objects according to the following local probability density measure:

$$f(\mathbf{X}_i) = \max\{0, \frac{1}{s^2} \sum_{\mathbf{X}_j \in N_{s\times s}(r)} [1 - \frac{d(\mathbf{X}_i, \mathbf{X}_j)}{\alpha(1 + \frac{v-1}{v_{\max}})} \tag{17}$$

In the above expression, $N_{s\times s}(r)$denotes the local area of perception surrounding the site of radius r, which the ant occupies in the two-dimensional grid. The threshold a scales the dissimilarity within each pair of objects, and the moving speed v controls the step-size of the ant searching in the space within one time unit. If an ant is not carrying an object and finds an object $\mathbf{X}_i$ in its neighborhood, it picks up this object with a probability that is inversely proportional to the number of similar objects in the neighborhood. It may be expressed as:

$$P_{pick-up}(\mathbf{X}_i) = [\frac{k_p}{k_p + f(\mathbf{X}_i)}]^2 \tag{18}$$

If however, the ant is carrying an object x and perceives a neighbor's cell in which there are other objects, then the ant drops off the object it is carrying with a probability that is directly proportional to the object's similarity with the perceived ones. This is given by:

$$\begin{aligned} P_{drop}(\vec{X}_i) &= 2.f(\vec{X}_i) \quad \text{if} \quad f(X_i) < k_d \\ &= \quad 1 \qquad\quad \text{if} \quad f(\vec{X}_i) \geq k_d \end{aligned} \tag{19}$$

The parameters k_p and k_d are the picking and dropping constants [41] respectively. Function $f(\mathbf{X}_i)$ provides an estimate of the density and similarity of elements in the neighborhood of object $\mathbf{X}_i$. The standard ACA algorithm is summarized in the following pseudo-code.

Procedure ACA

Place every item $\vec{X}_i$ on a random cell of the grid;
Place every ant k on a random cell of the grid unoccupied by ants;
iteration_count ← 1;
While iteration_count < maximum_iteration
do
for i = 1 to no_of_ants // for every ant
do
if unladen ant AND cell occupied by item $\vec{X}_i$,
then compute $f(\vec{X}_i)$ and $P_{pick-up}(\vec{X}_i)$;
pick up item $\vec{X}_i$ with probability $P_{pick-up}(\vec{X}_i)$
else if ant carrying item xi AND cell empty,
then compute $f(\vec{X}_i)$ and $P_{drop}(\vec{X}_i)$;
drop item $\vec{X}_i$ with probability $P_{drop}(\vec{X}_i)$;
end if
move to a randomly selected, neighboring and unoccupied cell ;
end for
t ← t + 1
end while
print location of items;
end procedure

Kanade and Hall (2003) presented a hybridization of the ant systems with the classical FCM algorithm to determine the number of clusters in a given dataset automatically. In their fuzzy ant algorithm, at first the ant based clustering is used to create raw clusters and then these clusters are refined using the FCM algorithm. Initially the ants move the individual data objects to form heaps. The centroids of these heaps are taken as the initial cluster centers and the FCM algorithm is used to refine these clusters. In the second stage the objects obtained from the FCM algorithm are hardened according to the maximum membership criteria to form new heaps. These new heaps are then sometimes moved and merged by the ants. The final clusters formed are refined by using the FCM algorithm.

A number of modifications have been introduced to the basic ant based clustering scheme that improve the quality of the clustering, the speed of convergence and, in particular, the spatial separation between clusters on the grid, which is essential for the scheme of cluster retrieval. A detailed description of the variants and results on the qualitative performance gains afforded by these extensions are provided in (Tsang and Kwong, 2006).

4.2 The PSO-based Clustering Algorithms

Research efforts have made it possible to view data clustering as an optimization problem. This view offers us a chance to apply PSO algorithm for evolving a set of candidate cluster

centroids and thus determining a near optimal partitioning of the dataset at hand. An important advantage of the PSO is its ability to cope with local optima by maintaining, recombining and comparing several candidate solutions simultaneously. In contrast, local search heuristics, such as the simulated annealing algorithm (Selim and Alsultan, 1991) only refine a single candidate solution and are notoriously weak in coping with local optima. Deterministic local search, which is used in algorithms like the k-means, always converges to the nearest local optimum from the starting position of the search.

PSO-based clustering algorithm was first introduced by Omran *et al.* (2002). The results of Omran *et al.* (2002, 2005) showed that PSO based method outperformed k-means, FCM and a few other state-of-the-art clustering algorithms. In their method, Omran *et al.* used a quantization error based fitness measure for judging the performance of a clustering algorithm. The quantization error is defined as:

$$J_e = \frac{\sum_{i=1}^{k} \sum_{\forall \mathbf{X}_j \in C_i} d(\mathbf{X}_j, \mathbf{V}_i)/n_i}{k} \tag{20}$$

where *Ci* is the *i*-th cluster center and *ni* is the number of data points belonging to the *i*-th cluster. Each particle in the PSO algorithm represents a possible set of *k* cluster centroids as:

$\vec{Z}_i(t)$ = | $\vec{V}_{i,1}$ | $\vec{V}_{i,2}$ | ..... | $\vec{V}_{i,k}$ |

where $\mathbf{V}_{i,p}$ refers to the *p*-th cluster centroid vector of the *i*-th particle. The quality of each particle is measured by the following fitness function:

$$f(\mathbf{Z}_i, M_i) = w_1 \bar{d}_{\max}(M_i, \mathbf{X}_i) + w_2(R_{\max} - d_{\min}(\mathbf{Z}_i)) + w_3 J_e \tag{21}$$

In the above expression, Rmax is the maximum feature value in the dataset and ***Mi*** is the matrix representing the assignment of the patterns to the clusters of the i-th particle. Each element mi, k, p indicates whether the pattern $\mathbf{X}_p$ belongs to cluster C_k of *i*-th particle. The user-defined constants *w*1, *w*2, and *w*3 are used to weigh the contributions from different sub-objectives. In addition,

$$\bar{d}_{\max} = \max_{j \in 1,2,\ldots,k} \{ \sum_{\forall \mathbf{X}_p \in C_{i,j}} d(\mathbf{X}_p, \mathbf{V}_{i,j})/n_{i,j} \} \tag{22}$$

and,

$$d_{\min}(\mathbf{Z}_i) = \min_{\forall p,q,p \neq q} \{ d(\mathbf{V}_{i,p}, \mathbf{V}_{i,q}) \} \tag{23}$$

is the minimum Euclidean distance between any pair of clusters. In the above, *ni*,*k* is the number of patterns that belong to cluster **C***i*,*k* of particle *i*. he fitness function is a multi-objective optimization problem, which minimizes the intra-cluster distance, maximizes inter-cluster separation, and reduces the quantization error. The PSO clustering algorithm is summarized below.

Step 1: Initialize each particle with k random cluster centers.
Step 2: repeat for iteration_count = 1 to maximum_iterations
(a) repeat for each particle i
(i) repeat for each pattern $\vec{X}_p$ in the dataset
- calculate Euclidean distance of $\vec{X}_p$ with all cluster centroids.
- assign $\vec{X}_p$ to the cluster that have nearest centroid to $\vec{X}_p$

(ii) calculate the fitness function $f(\vec{Z}_i, M_i)$
(b) find the personal best and global best position of each particle.
(c) Update the cluster centroids according to velocity updating and coordinate updating formula of PSO.

Van der Merwe and Engelbrecht hybridized this approach with the k-means algorithm for clustering general dataets (van der Merwe and Engelbrecht, 2003). A single particle of the swarm is initialized with the result of the k-means algorithm. The rest of the swarm is initialized randomly. In 2003, Xiao *et al* used a new approach based on the synergism of the PSO and the Self Organizing Maps (SOM) (Xiao *et al*., 2003) for clustering gene expression data. They got promising results by applying the hybrid SOM-PSO algorithm over the gene expression data of Yeast and Rat Hepatocytes. Paterlini and Krink (2006) have compared the performance of k-means, GA (Holland, 1975, Goldberg, 1975), PSO and Differential Evolution (DE) (Storn and Price, 1997) for a representative point evaluation approach to partitional clustering. The results show that PSO and DE outperformed the k-means algorithm.

Cui et al. (2005) proposed a PSO based hybrid algorithm for classifying the text documents. They applied the PSO, k-means and a hybrid PSO clustering algorithm on four different text document datasets. The results illustrate that the hybrid PSO algorithm can generate more compact clustering results over a short span of time than the k-means algorithm.

5 Automatic Kernel-based Clustering with PSO

The Euclidean distance metric, employed by most of the exisiting partitional clustering algorithms, work well with datasets in which the natural clusters are nearly hyper-spherical and linearly seperable (like the artificial dataset 1 used in this paper). But it causes severe misclassifications when the dataset is complex, with linearly non-separable patterns (like the synthetic datasets 2, 3, and 4 described in Section 5.8.1 of the chapter). We would like to mention here that, most evolutionary algorithms could potentially work with an arbitrary distance function and are not limited to the Euclidean distance.

Moreover, very few works (Bandyopadhyay and Maulik, 2000, Rosenberger and Chehdi, 2000, Omran *et al*., 2005, Sarkar *et al*., 1997) have been undertaken to make an algorithm learn the correct number of clusters 'k' in a dataset, instead of accepting the same as a user input. Although, the problem of finding an optimal k is quite important from a practical point

of view, the research outcome is still unsatisfactory even for some of the benchmark datasets (Rosenberger and Chehdi, 2000).

In this Section, we describe a new approach towards the problem of automatic clustering (without having any prior knowledge of k initially) in kernel space using a modified version of the PSO algorithm (Das *et al*., 2008). Our procedure employs a kernel induced similarity measure instead of the conventional Euclidean distance metric. A kernel function measures the distance between two data points by implicitly mapping them into a high dimensional feature space where the data is linearly separable. Not only does it preserve the inherent structure of groups in the input space, but also simplifies the associated structure of the data patterns (Girolami, 2002). Several kernel-based learning methods, including the Support Vector Machine (SVM), have recently been shown to perform remarkably in supervised learning (Scholkopf and Smola, 2002, Vapnik, 1998, Zhang and Chen, 2003, Zhang and Rudnicky, 2002). The kernelized versions of the k-means and the fuzzy c-means (FCM) algorithms reported in (Zhang and Rudnicky, 2002) and (Zhang and Chen, 2003) respectively, have reportedly outperformed their original counterparts over several test cases.

Now, we may summarize the new contributions presented here in the following way:

1. Firstly, we develop an alternative framework for learning the number of partitions in a dataset besides the simultaneous refining of the clusters, through one shot of optimization.
2. We propose a new version of the PSO algorithm based on the multi-elitist strategy, well-known in the field of evolutionary algorithms. Our experiments indicate that the proposed MEPSO algorithm yields more accurate results at a faster pace than the classical PSO in context to the present problem.
3. We reformulate a recently proposed cluster validity index (known as the *CS measure* (Chou *et al*., 2004)) using the kernelized distance metric. This reformulation eliminates the need to compute the cluster-centroids repeatedly for evaluating CS value, due to the implicit mapping via the kernel function. The new CS measure forms the objective function to be minimized for optimal clustering.

5.1 The Kernel Based Similarity Measure

Given a dataset $\mathbf{X}$ in the d-dimensional real space $\Re^d$, let us consider a non-linear mapping function from the input space to a high dimensional feature space H:

$$\varphi : \Re^d \to H, \mathbf{x}_i \to \varphi(\mathbf{x}_i) \tag{24}$$

where $\mathbf{x}_i = [x_{i,1}, x_{i,2}, ..., x_{i,d}]^T$ and

$$\varphi(\mathbf{x}_i) = [\varphi_1(\mathbf{x}_i), \varphi_2(\mathbf{x}_i), ..., \varphi_H(\mathbf{x}_i)]^T$$

By applying the mapping, a dot product $\mathbf{x}_i^T.\mathbf{x}_j$is transformed into $\varphi^T(\mathbf{x}_i).\varphi(\mathbf{x}_j)$. Now, the central idea in kernel-based learning is that the mapping function φneed not be explicitly specified. The dot product $\varphi^T(\mathbf{x}_i).\varphi(\mathbf{x}_j)$in the transformed space can be calculated through the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ in the input space $\Re^d$. Consider the following simple example:
Example 1: let d = 2 and H = 3 and consider the following mapping:
$\varphi : \Re^2 \to H = \Re^3$, and $[x_{i,1}, x_{i,2}]^T \to [x_{i,1}^2, \sqrt{2}.x_{i,1}x_{i,2}, x_{i,2}^2]^T$
Now the dot product in feature space H:

$$\varphi^T(\mathbf{x}_i).\varphi(\mathbf{x}_j) = [x_{i,1}^2, \sqrt{2}.x_{i,1}x_{i,2}, x_{i,2}^2].[x_{j,1}^2, \sqrt{2}.x_{j,1}.x_{j,2}, x_{j,2}^2]^T.$$

$$= [x_{i,1}.x_{j,1} + x_{i,2}.x_{j,2}]^2$$

$$= [\mathbf{x}_i^T.\mathbf{x}_j]^2 = K(\mathbf{x}_i, \mathbf{x}_j)$$

Clearly the simple kernel function K is the square of the dot product of vectors $\mathbf{x}_i$ and $\mathbf{x}_j$in $\Re^d$. Hence, the kernelized distance measure between two patterns $\mathbf{x}_i$ and $\mathbf{x}_j$is given by:

$$\left\|\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)\right\|^2 = (\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j))^T(\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j))$$

$$= \varphi^T(\mathbf{x}_i).\varphi(\mathbf{x}_i) - 2.\varphi^T(\mathbf{x}_i).\varphi(\mathbf{x}_j) + \varphi^T(\mathbf{x}_j).\varphi(\mathbf{x}_j)$$

$$= K(\mathbf{x}_i, \mathbf{x}_i) - 2.K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_j, \mathbf{x}_j) \tag{25}$$

Among the various kernel functions used in literature, in the present context, we have chosen the well-known Gaussian kernel (also referred to as the Radial Basis Function) owing to its better classification accuracy over the linear and polynomial kernels on many test problems (Pirooznia and Deng, 2006, Hertz *et al*., 2006). The Gaussian Kernel may be represented as:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\sigma^2}\right) \tag{26}$$

where $\sigma > 0$. Clearly, for Gaussian kernel, $K(\mathbf{x}_i, \mathbf{x}_i)$= 1 and thus relation 25 reduces to:

$$\left\|\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_j)\right\|^2 = 2.(1 - K(\mathbf{x}_i, \mathbf{x}_j)) \tag{27}$$

5.2 Reformulation of CS Measure

Cluster validity indices correspond to the statistical-mathematical functions used to evaluate the results of a clustering algorithm on a quantitative basis. For crisp clustering, some of the well-known indices available in the literature are the Dunn's index (DI) (Dunn, 1974), Calinski-Harabasz index (Calinski and Harabasz, 1975), Davis-Bouldin (DB) index (Davies and Bouldin, 1979), PBM index (Pakhira *et al*., 2004), and the CS measure (Chou *et al*., 2004). In this work, we have based our fitness function on the CS measure as according to the authors, CS measure is more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures, the price being paid in terms of high computational load with increasing k and n (Chou *et al*., 2004). Before applying the CS measure, centroid of a cluster is computed by averaging the data vectors belonging to that cluster using the formula,

$$\mathbf{m}_i = \frac{1}{N_i}\sum_{x_j \in C_i} \mathbf{x}_j \tag{28}$$

A distance metric between any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is denoted by $d(\mathbf{x}_i, \mathbf{x}_j)$. Then the CS measure can be defined as,

$$CS(k) = \frac{\frac{1}{k}\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\mathbf{X}_i \in C_i}\max_{\mathbf{X}_q \in C_i}\{d(\mathbf{x}_i, \mathbf{x}_q)\}]}{\frac{1}{k}\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{d(\mathbf{m}_i, \mathbf{m}_j)\}]} = \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\mathbf{X}_i \in C_i}\max_{\mathbf{X}_q \in C_i}\{d(\mathbf{x}_i, \mathbf{x}_q)\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{d(\mathbf{m}_i, \mathbf{m}_j)\}]} \tag{29}$$

Now, using a Gaussian kernelized distance measure and transforming to the high dimensional feature space, the CS measure reduces to (using relation (23)):

$$CS_{\ker nel}(k) = \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\mathbf{X}_i \in C_i} \max_{\mathbf{X}_q \in C_i}\{||\varphi(\mathbf{x}_i) - \varphi(\mathbf{x}_q)||^2\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{||\varphi(\mathbf{m}_i) - \varphi(\mathbf{m}_j)||\}]}$$
$$= \frac{\sum_{i=1}^{k}[\frac{1}{N_i}\sum_{\mathbf{X}_i \in C_i} \max_{\mathbf{X}_q \in C_i}\{2(1 - K(\mathbf{x}_i, \mathbf{x}_q))\}]}{\sum_{i=1}^{k}[\min_{j \in K, j \neq i}\{2(1 - K(\mathbf{m}_i, \mathbf{m}_j))\}]}$$

The minimum value of this CS measure indicates an optimal partition of the dataset. The value of 'k' which minimizes CSkernel(k) therefore gives the appropriate number of clusters in the dataset.

5.3 The Multi-Elitist PSO (MEPSO) Algorithm

The canonical PSO has been subjected to empirical and theoretical investigations by several researchers (Eberhart and Shi, 2001, Clerc and Kennedy, 2002). In many occasions, the convergence is premature, especially if the swarm uses a small inertia weight ω or constriction coefficient (Eberhart and Shi, 2001). As the global best found early in the searching process may be a poor local minima, we propose a multi-elitist strategy for searching the global best of the PSO. We call the new variant of PSO the MEPSO. The idea draws inspiration from the works reported in (Deb *et al*., 2002). We define a growth rate β for each particle. When the fitness value of a particle at the t-th iteration is higher than that of a particle at the (t-1)-th iteration, the β will be increased. After the local best of all particles are decided in each generation, we move the local best, which has higher fitness value than the global best into the candidate area. Then the global best will be replaced by the local best with the highest growth rate β. The elitist concept can prevent the swarm from tending to the global best too early in the searching process. The MEPSO follows the *g_best* PSO topology in which the entire swarm is treated as a single neighborhood. The pseudo code about MEPSO is as follows:

Procedure MEPSO
For t =1 to $t_{\max}$
  For j =1 to N *// swarm size is N*
    If (the fitness value of particle$_j$ in t-th time-step> that of particle$j$ in (t-1)-th time-step)
      $\beta_j(t) = \beta_j(t\text{-}1) + 1;$
    End
    Update Local best$j$.
    If (the fitness of Local bestj > that of Global best now*)*
      Choose Local bestj put into candidate area.
    End
  End
  Calculate β of every candidate, and record the candidate of $\beta_{\max}$.
  Update the Global best to become the candidate of $\beta_{\max}$.
  Else
    Update the Global best to become the particle of highest fitness value.
  End
End

5.4 Particle Representation

In the proposed method, for n data points, each p-dimensional, and for a user-specified maximum number of clusters $k$max, a particle is a vector of real numbers of dimension $k$max + $k$max $\times$ p. The first $k$max entries are positive floating-point numbers in (0, 1), each of which determines whether the corresponding cluster is to be activated (i.e. to be really used for classifying the data) or not. The remaining entries are reserved for kmax cluster centers, each p-dimensional.

A single particle is illustrated as:

$\vec{Z}_i(t) =$ | $T_{i,1}$ | $T_{i,2}$ | ..... | $T_{i,k\max}$ | $\vec{m}_{i,1}$ | $\vec{m}_{i,2}$ | ..... | $\vec{m}_{i,k_{\max}}$ |

Activation Threshhold **Cluster Centroids**

The j-th cluster center in the i-th particle is active or selected for partitioning the associated dataset if Ti,j $>$ 0.5. On the other hand, if Ti,j $<$ 0.5, the particular j-th cluster is inactive in the i-th particle. Thus the Ti,j s behave like control genes (we call them *activation thresholds*) in the particle governing the selection of the active cluster centers. The rule for selecting the actual number of clusters specified by one chromosome is:

IF $T_{i,j} > 0.5$ **THEN** the j-th cluster center $\vec{m}_{i,j}$ is **ACTIVE**

ELSE $\vec{m}_{i,j}$ is **INACTIVE** (30)

Consider the following example:
Example 2: Positional coordinates of one particular particle is illustrated below. There are at most five 3-dimensional cluster centers among which, according to the rule presented in Equation 30 the second (6, 4.4, 7), third (5.3, 4.2, 5) and fifth one (8, 4, 4) have been activated for partitioning the dataset and marked in bold. The quality of the partition yielded by such a particle can be judged by an appropriate cluster validity index.

0.3	**0.6**	**0.8**	0.1	**0.9**	6.1	3.2	2.1	**6**	**4.4**	**7**	**9.6**	**5.3**	**4.2**	5	8	4.6	**8**	**4**	**4**

During the PSO iterations, if some threshold T in a particle exceeds 1 or becomes negative, it is fixed to 1 or zero, respectively. However, if it is found that no flag could be set to one in a particle (all activation threshholds are smaller than 0.5), we randomly select 2 thresholds and re-initialize them to a random value between 0.5 and 1.0. Thus the minimum number of possible clusters is 2.

5.5 The Fitness Function

One advantage of the proposed algorithm is that it can use any suitable validity index as its fitness function. We have used the kernelized CS measure as the basis of our fitness function, which for i-th particle can be described as:

$$f_i = \frac{1}{CS_{\mathrm{ker}nel_i}(k) + eps} \tag{31}$$

where *eps* is a very small constant (we used 0.0002). Maximization of fi implies a minimization of the kernelized CS measure leading to the optimal partitioning of the dataset.

5.6 Avoiding Erroneous particles with Empty Clusters or Unreasonable Fitness Evaluation

There is a possibility that in our scheme, during computation of the kernelized CS index, a division by zero may be encountered. For example, the positive infinity (such as 4.0/0.0) or the not-a-number (such as 0.0/0.0) condition always occurs when one of the selected cluster centers is outside the boundary of distributions of data set as far as possible. To avoid this problem we first check to see if in any particle, any cluster has fewer than 2 data points in it. If so, the cluster center positions of this special particle are re-initialized by an average computation. If k clusters ($2 \leq k \leq k_{\max}$) are selected for this particle, we put n/k data points for every individual activated cluster center, such that a data point goes with a center that is nearest to it.

5.7 Putting It All Together

Step 1: Initialize each particle to contain k number of randomly selected cluster centers and k (randomly chosen) activation thresholds in [0, 1].

Step 2: Find out the active cluster centers in each particle with the help of the rule described in (10).

Step 3: For t =1to $t_{\max}$ **do**

i) For each data vector $\vec{X}_p$, calculate its distance metric $d(\vec{X}_p, \vec{m}_{i,j})$ from all active cluster centers of the i-th particle $\vec{V}_i$.

ii) Assign $\vec{X}_p$ to that particular cluster center $\vec{m}_{i,j}$ where

$$d(\vec{X}_p, \vec{m}_{i,j}) = \min_{\forall b \in \{1,2,\ldots,k\}} \left\{ d(\vec{X}_p, \vec{m}_{i,b}) \right\}$$

iii) Check if the number of data points belonging to any cluster center $\vec{m}_{i,j}$ is less than 2. If so, update the cluster centers of the particle using the concept of average described earlier.

iv) Change the population members according to the MEPSO algorithm. Use the fitness of the particles to guide the dynamics of the swarm.

Step 4: Report as the final solution the cluster centers and the partition obtained by the globally best particle (one yielding the highest value of the fitness function) at time $t = t_{\max}$.

5.8 Experimental Results

Comparison with Other Clustering Algorithms

To test the effectiveness of the proposed method, we compare its performance with six other clustering algorithms using a test-bed of five artificial and three real world datasets. Among the competitors, there are two recently developed automatic clustering algorithms well- known as the GCUK (Genetic Clustering with an Unknown number of clusters *k*) (Bandyopadhyay and Maulik, 2000) and the DCPSO (Dynamic Clustering PSO) (Omran *et al*., 2005). Moreover, in order to investigate the effects of the changes made in the classical *g_best* PSO algorithm, we have compared MEPSO with an ordinary PSO based kernel-clustering method that uses the same particle representation scheme and fitness function as the MEPSO. Both the algorithms were let run on the same initial populations. The rest of the competitors are the kernel k-means algorithm (Zhang and Rudnicky, 2002) and a kernelized version of the subtractive clustering (Kim *et al*., 2005). Both the algorithms were provided with the correct number of clusters as they are non-automatic.

We used datasets with a wide variety in the number and shape of clusters, number of datapoints and the count of features of each datapoint. The real life datasets used here are the Glass, the placeWisconsin breast cancer, the image segmentation, the Japanese Vowel and the automobile (Handl *et al*., 2003). The synthetic datasets included here, comes with linearly

non-separable clusters of different shapes (like elliptical, concentric circular dish and shell, rectangular etc). Brief details of the datasets have been provided in Table 1. Scatterplot of the synthetic datasets have also been shown in Figure 5.The clustering results were judged using Huang's accuracy measure (Huang and Ng, 1999):

$$r = \frac{\sum_{i=1}^{k} n_i}{n}, \tag{32}$$

where n_i is the number of data occurring in both the i-th cluster and its corresponding true cluster, and n is the total number of data points in the data set. According to this measure, a higher value of r indicates a better clustering result, with perfect clustering yielding a value of $r = 1$.

We used $\sigma = 1.1$ for all the artificial datasets, $\sigma = 0.9$ for breast cancer dataset and $\sigma = 2.0$ for the rest of the real life datasets for the RBF kernel following (Lumer and Faieta, 1995).In these experiments, the kernel k-means was run 100 times with the initial centroids randomly selected from the data set. A termination criterion of $\varepsilon = 0.001$. The parameters of the kernel-based subtractive methods were set to $\alpha = 5.4$ and $\beta = 1.5$ as suggested by Pal and Chakraborty (Kuntz and Snyers, 1994). For all the competitive algorithms, we have selected their best parameter settings as reported in the corresponding literatures. The control parameters for MEPSO where chosen after experimenting with several possible values. Some of the experiments focussing on the effects of parameter-tuning in MEPSO has been reported in the next subsection. The same set of parameters were used for all the test problems for all the algorithms. These parameter settings have been reported in Table 2.

Table 3 compares the algorithms on the quality of the optimum solution as judged by the Huang's measure. The mean and the standard deviation (within parentheses) for 40 independent runs (with different seeds for the random number generator) of each of the six algorithms are presented in Table 3. Missing values of standard deviation in this table indicate a zero standard deviation. The best solution in each case has been shown in bold. Table 4 and 5 present the mean and standard deviation of the number of classes found by the three automatic clustering algorithms. In Figure 2 we present the clustering results on the synthetic datasets by the new MEPSO algorithm (to save space we do not provide results for all the six algorithms).

For comparing the speed of the stochastic algorithms like GA, PSO etc. we choose *number of fitness function evaluations* (placeFEs) as a measure of computation time instead of generations or iterations. From the data provided in Table 3, we choose a threshold value of the classification accuracy for each dataset. This threshold value is somewhat larger than the minimum accuracy attained by each automatic clustering algorithm. Now we run an algorithm on each dataset and stop as soon as it achieves the proper number of clusters as well as the threshold accuracy. We then note down the number of fitness function evaluations the algorithm takes. A lower number of placeFEs corresponds to a faster algorithm. The speed comparison results are provided in Table 6. The kernel k-means and the subtractive clustering method are not included in this table, as they are non-automatic and do not employ evolutionary operators as in GCUK and PSO based methods.

Table 1: Description of the Datasets

Dateset	Number of Datapoints	Number of clusters	Data-dimension
Synthetic_1	500	2	2
Synthetic_2	52	2	2
Synthetic_3	400	4	3
Synthetic_4	250	5	2
Synthetic_5	600	2	2
Glass	214	6	9
Wine	178	3	13
Breast Cancer	683	2	9
Image Segmentation	2310	7	19
Japanese Vowel	640	9	12

Table 2: Parameter Settings for different algorithms

GCUK		DCPSO		PSO		MEPSO	
Pop_size	70	Pop_size	100	Pop_ size	40	Pop_ size	40
Cross-over Probability μ_c	0.85	Inertia Weight	0.72	Inertia Weight	0.75	Inertia Weight	0.794
Mutation probability μ_m	0.005	C_1, C_2	1.494	C_1, C_2	2.00	C_1, C_2	0.35→2.4 2.4→0.35
		P_{ini}	0.75				
K_{max}	20	K_{max}	20	K_{max}	20	K_{max}	20
K_{min}	2	K_{min}	2	K_{min}	2	K_{min}	2

A review of Tables 3, 4 and 5 reveals that the kernel based MEPSO algorithm performed markedly better as compared to the other considered clustering algorithms, in terms of both accuracy and convergence speed. We note that in general, the kernel based clustering methods outperform the GCUK or DCPSO algorithms (which do not use the kernelized fitness function) especially on linearly non-separable artificial datasets like synthetic_1, synthetic_2 and synthetic_5. Although the proposed method provided a better clustering result than the other methods for Synthetic_5 dataset, its accuracy for this data was lower than the seven other data sets considered. This indicates that the proposed approach is limited in its ability to classify non-spherical clusters.

The PSO based methods (especially MEPSO) on average took lesser computational time than the GCUK algorithm over most of the datasets. One possible reason of this may be the use of less complicated variation operators (like mutation) in PSO as compared to the operators used for GA. We also note that the MEPSO performs much better than the classical PSO based kernel-clustering scheme. Since both the algorithms use same particle representation and starts with the same initial population, difference in their performance must be due to the difference in their internal operators and parameter values. This demonstrates the effectiveness of the multi-elitist strategy incorporated in the MEPSO algorithm

Table 3: Mean and standard deviation of the clustering accuracy (%) achieved by each clustering algorithm over 40 independent runs (Each run continued up to 50, 000 FEs for GCUK, DCPSO, Kernel_ PSO and Kernel_MEPSO)

Datasets	Algorithms					
	Kernel k-means	Kernel Sub_clust	GCUK	DC-PSO	Kernel_ PSO	Kernel_ MEPSO
Synthetic_1	83.45 (0.032)	87.28	54.98 (0.88)	57.84 (0.065)	90.56 (0.581)	**99.45 (0.005)**
Synthetic_2	71.32 (0.096)	75.73	65.82 (0.146)	59.91 (0.042)	61.41 (0.042)	**80.92 (0.0051)**
Synthetic_3	89.93 (0.88)	94.03	97.75 (0.632)	97.94 (0.093)	92.94 (0.193)	**99.31 (0.001)**
Synthetic_4	67.65 (0.104)	80.25	74.30 (0.239)	75.83 (0.033)	78.85 (0.638)	**87.84 (0.362)**
Synthetic_5	81.23 (0.127)	84.33	54.45 (0.348)	52.55 (0.209)	89.46 (0.472)	**99.75 (0.001)**
Glass	68.92 (0.032)	73.92	76.27 (0.327)	79.45 (0.221)	70.71 (0.832)	**92.01 (0.623)**
Wine	73.43 (0.234)	59.36	80.64 (0.621)	85.81 (0.362)	87.65 (0.903)	**93.17 (0.002)**
Breast Cancer	66.84 (0.321)	70.54	73.22 (0.437)	78.19 (0.336)	80.49 (0.342)	**86.35 (0.211)**
Image Segmentation	56.83 (0.641)	70.93	78.84 (0.336)	81.93 (1.933)	84.32 (0.483)	**87.72 (0.982)**
Japanese Vowel	44.89 (0.772)	61.83	70.23 (1.882)	82.57 (0.993)	79.32 (2.303)	**84.93 (2.292)**
Average	72.28	75.16	74.48	76.49	77.58	**91.65**

Table 4: Mean and standard deviation (in parenthesis) of the number of clusters found over the synthetic datasets for four automatic clustering algorithms over 40 independent runs.

Algorithms	Synthetic _1	Synthetic _2	Synthetic _3	Synthetic _4	Synthetic _5
GCUK	2.50 (0.021)	3.05 (0.118)	4.15 (0.039)	9.85 (0.241)	4.25 (0.921)
DCPSO	2.45 (0.121)	2.80 (0.036)	4.25 (0.051)	9.05 (0.726)	6.05 (0.223)
Ordinary PSO	2.50 (0.026)	2.65 (0.126)	4.10 (0.062)	9.35 (0.335)	2.25 (0.361)
Kernel_MEPSO	**2.10 (0.033)**	**2.15 (0.102)**	**4.00 (0.00)**	**10.05 (0.021)**	**2.05 (0.001)**

Table 5: Mean and standard deviation (in parenthesis) of the number of clusters found over the synthetic datasets for four automatic clustering algorithms over 40 independent runs.

Algorithms	Glass	Wine	Breast Cancer	Image Segmentation	Japanese Vowel
GCUK	5.85 (0.035)	4.05 (0.021)	2.25 (0.063)	7.05 (0.008)	9.50 (0.218)
DCPSO	5.60 (0.009)	3.75 (0.827)	2.25 (0.026)	7.50 (0.057)	10.25 (1.002)
Ordinary PSO	5.75 (0.075)	**3.00 (0.00)**	**2.00 (0.00)**	7.20 (0.025)	9.25 (0.822)
Kernel_MEPSO	**6.05 (0.015)**	**3.00 (0.00)**	**2.00 (0.00)**	**7.00 (0.00)**	**9.05 (0.021)**

Table 6: Mean and standard deviations of the number of fitness function evaluations (over 40 successful runs) required by each algorithm to reach a predefined cut-off value of the classification accuracy

Dateset	Threshold accuracy (in %)	GCUK	DCPSO	Ordinary PSO	Kernel_ MEPSO
Synthetic_1	50.00	48000.05 (21.43)	42451.15 (11.57)	43812.30 (2.60)	**37029.65 (17.48)**
Synthetic_2	55.00	41932.10 (12.66)	45460.25 (20.97)	40438.05 (18.52)	**36271.05 (10.41)**
Synthetic_3	85.00	40000.35 (4.52)	35621.05 (12.82)	37281.05 (7.91)	**32035.55 (4.87)**
Synthetic_4	65.00	46473.25 (7.38)	43827.65 (2.75)	42222.50 (2.33)	**36029.05 (6.38)**
Synthetic_5	50.00	43083.35 (5.30)	39392.05 (7.20)	42322.05 (2.33)	**35267.45 (9.11)**
Glass	65.00	47625.35 (6.75)	40382.15 (7.27)	38292.25 (10.32)	**37627.05 (12.36)**
Wine	55.00	44543.70 (44.89)	43425.00 (18.93)	3999.65 (45.90)	**35221.15 (67.92)**
Breast Cancer	65.00	40390.00 (11.45)	37262.65 (13.64)	35872.05 (8.32)	**32837.65 (4.26)**
Image Segmentation	55.00	39029.05 (62.09)	40023.25 (43.92)	35024.35 (101.26)	**34923.22 (24.28)**
Japanese Vowel	40.00	40293.75 (23.33)	28291.60 (121.72)	29014.85 (21.67)	**24023.95 (20.62)**

Choice of Parameters for MEPSO

The MEPSO has a number of control parameters that affect its performance on different clustering problems. In this Section we discuss the influence of parameters like swarm size, the inertia factor ω and the acceleration factors C1 and C2 on the Kernel_MEPSO algorithm.

1. **Swarm Size**: To investigate the effect of the swarm size, the MEPSO was executed separately with 10 to 80 particles (keeping all other parameter settings same as reported in

Table 2) on all the datasets. In Figure 6 we plot the convergence behavior of the algorithm (average of 40 runs) on the image segmentation dataset (with 2310 data points and 19 features, it is the most complicated synthetic dataset considered here) for different population sizes. We omit the other results here to save space. The results reveal that the number of particles more than 40 gives more or less identical accuracy of the final clustering results for MEPSO. This observation is in accordance with Van den Bergh and Engelbrecht, who in (van den Bergh and Engelbrecht, 2001) showed that though there is a slight improvement in the solution quality with increasing swarm sizes, a larger swarm increases the number of function evaluations to converge to an error limit. For most of the problems, it was found that keeping the swarm size 40 provides a reasonable trade-off between the quality of the final results and the convergence speed.

2. **The inertia factor ω:** Provided all other parameters are fixed at the values shown in Table 2, the MEPSO was run with several possible choices of the inertia factor ω. Specifically we used a time-varying ω (linearly decreasing from 0.9 to 0.4 following (Shi and Eberhart, 1999)), random ω (Eberhart and Shi, 2001),ω = 0.1, ω =0.5 and finally ω = 0.794 (Kennedy *et al*., 2001). In Figure 7, we show how the fitness of the globally best particle (averaged over 40 runs) varies with no. of placeFEs for the image segmentation dataset over different values of ω. It was observed that for all the problems belonging to the current test suit, best convergence behavior of MEPSO is observed for ω = 0.794.
3. **The acceleration coefficients $C1$ and $C2$:** Provided all other parameters are fixed at the values given in Table 2, we let MEPSO run for different settings of the acceleration coefficients $C1$ and $C2$ as reported in various literatures on PSO. We used $C1$ = 0.7 and $C2$ = 1.4, $C1$ = 1.4 and $C2$ = 0.7 (Shi and Eberhart, 1999), $C1$ = $C2$ = 1.494 (Kennedy *et al*., 2001), $C1$=$C2$=2.00 (Kennedy *et al*., 2001) and finally a time varying acceleration coefficients where $C1$ linearly increases from 0.35 to 2.4 and $C2$ linearly decreases from 2.4 to 0.35 (Ratnaweera and Halgamuge, 2004). We noted that the linearly varying acceleration coefficients gave the best clustering results over all the problems considered. This is perhaps due to the fact that an increasing $C1$ and gradually decreasing $C2$ boost the global search over the entire search space during the early part of the optimization and encourage the particles to converge to global optima at the end of the search. Figure 8 illustrates the convergence characteristics of MEPSO over the image segmentation dataset for different settings of $C1$ and $C2$.

6 Conclusion and Future Directions

In this Chapter, we introduced some of the preliminary concepts of Swarm Intelligence (SI) with an emphasis on particle swarm optimization and ant colony optimization algorithms. We then described the basic data clustering terminologies and also illustrated some of the past and ongoing works, which apply different SI tools to pattern clustering problems. We proposed a novel kernel-based clustering algorithm, which uses a deviant variety of the PSO. The proposed algorithm can automatically compute the optimal number of clusters in any dataset and thus requires minimal user intervention. Comparison with a state of the art GA based clustering strategy, reveals the superiority of the MEPSO-clustering algorithm both in terms of accuracy and speed.

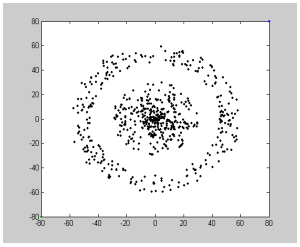
Despite being an age old problem, clustering remains an active field of interdisciplinary research till date. No single algorithm is known, which can group all real world datasets efficiently and without error. To judge the quality of a clustering, we need some specially designed statistical-mathematical function called the clustering validity index. But a literature

survey reveals that, most of these validity indices are designed empirically and there is no universally good index that can work equally well over any dataset. Since, majority of the PSO or ACO based clustering schemes rely on a validity index to judge the fitness of several possible partitioning of the data, research effort should be spent for defining a reasonably good index function and validating the same mathematically.

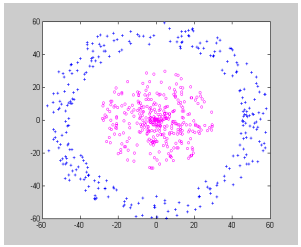
Feature extraction is an important preprocessing step for data clustering. Often we have a great number of features (especially for a high dimensional dataset like a collection of text documents) which are not all relevant for a given operation. Hence, future research may focus on integrating the automatic feature-subset selection scheme with the SI based clustering algorithm. The two-step process is expected to automatically project the data to a low dimensional feature subspace, determine the number of clusters and find out the appropriate cluster centers with the most relevant features at a faster pace.

Gene expression refers to a process through which the coded information of a gene is converted into structures operating in the cell. It provides the physical evidence that a gene has been ”turned on” or activated for protein synthesis (Lewin, 1995). Proper selection, analysis and interpretation of the gene expression data can lead us to the answers of many important problems in experimental biology. Promising results have been reported in (Xiao *et al*., 2003) regarding the application of PSO for clustering the expression levels of gene subsets. The research effort to integrate SI tools in the mechanism of gene expression clustering may in near future open up a new horizon in the field of bioinformatic data mining.

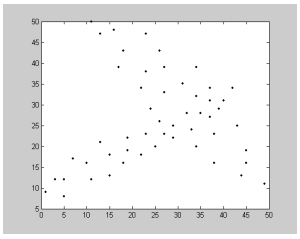
Hierarchical clustering plays an important role in fields like information retrieval and web mining. The self-assembly behavior of the real ants may be exploited to build up new hierarchical tree-structured partitioning of a data set according to the similarities between those data items. A description of the little but promising work already been undertaken in this direction can be found in (Azzag *et al*., 2006). But a more extensive and systematic research effort is necessary to make the ant based hierarchical models superior to existing algorithms like Birch (Zhang *et al*., 1997).



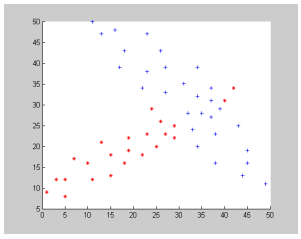
(a) Unlabeled Synthetic_1



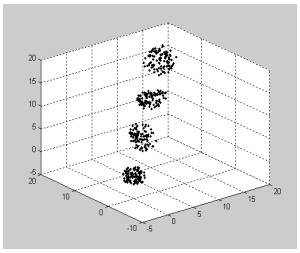
Synthetic_1 Clustered with Kernel_MEPSO



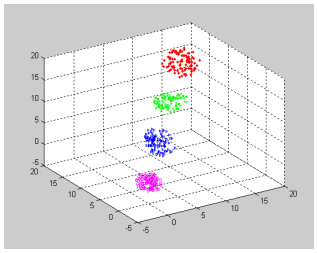
(b) Unlabeled Synthetic_2



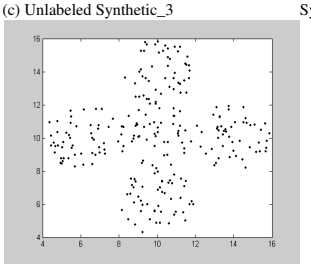
Synthetic_2 Clustered with Kernel_MEPSO



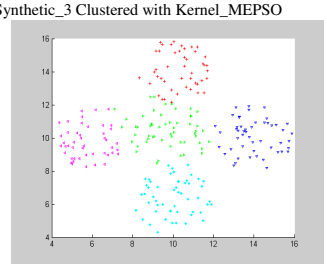
(c) Unlabeled Synthetic_3



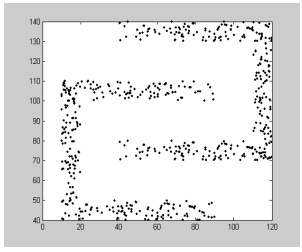
Synthetic_3 Clustered with Kernel_MEPSO



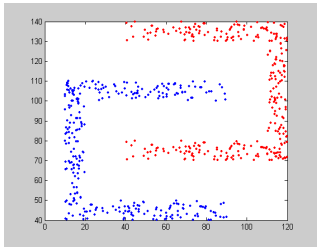
(d) Unlabeled Synthetic_4



Synthetic_4 Clustered with Kernel_MEPSO



(e) Unlabeled Synthetic_5



Synthetic_5 Clustered with Kernel_MEPSO

Fig. 5. Two- and three-dimensional synthetic datasets clustered with MEPSO.

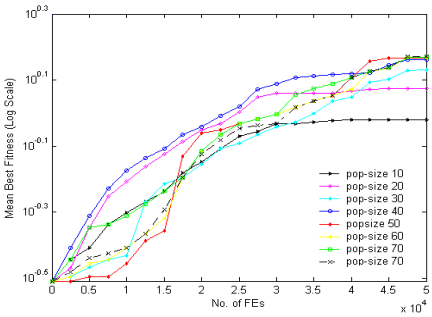


Fig. 6. The convergence characteristics of the MEPSO over the Image segmentation dataset for different population sizes.

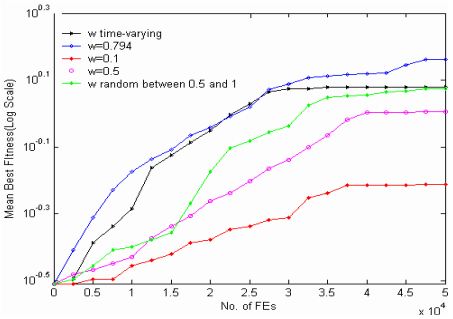


Fig. 7. The convergence characteristics of the MEPSO over the Image segmentation dataset for different inertia factors.

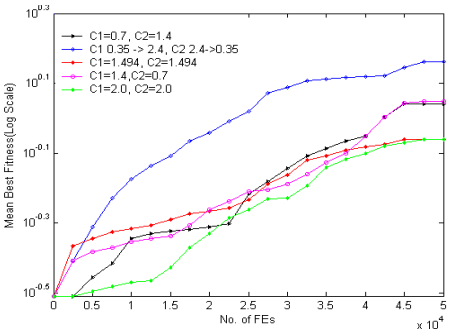


Fig. 8. The convergence characteristics of the MEPSO over the Image segmentation dataset for different acceleration coefficients.

References

A. Abraham, C. Grosan and V. Ramos (Eds.) (2006), Swarm Intelligence and Data Mining, Studies in Computational Intelligence, Springer Verlag, Germany, pages 270, ISBN: 3-540-34955-3.

Ahmed MN, Yaman SM, Mohamed N, Farag AA and Moriarty TA (2002) Modified fuzzy c-means algorithm for bias field estimation and segmentation of MRI data. IEEE Trans Med Imaging, 21, pp. 193199.

Azzag H, Guinot C and Venturini G (2006) Data and text mining with hierarchical clustering ants, in Swarm Intelligence in Data Mining, Abraham A, Grosan C and Ramos V (Eds), Springer, pp. 153-186.

Bandyopadhyay S and Maulik U (2000) Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, 35, pp. 1197-1208.

Beni G and Wang U (1989) Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robots and Biological Systems, Il Ciocco, Tuscany, Italy.

Bensaid AM, Hall LO, Bezdek JC.and Clarke LP (1996) Partially supervised clustering for image segmentation. Pattern Recognition, vol. 29, pp. 859-871.

Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. New York: Plenum.

Bonabeau E, Dorigo M and Theraulaz G (1999) Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York.

Brucker P (1978) On the complexity of clustering problems. Beckmenn M and Kunzi HP(Eds.), Optimization and Operations Research, Lecture Notes in Economics and Mathematical Systems, Berlin, Springer, vol.157, pp. 45-54.

Calinski RB and Harabasz J (1975) Adendrite method for cluster analysis, Commun. Statistics, 1 27.

Chou CH, Su MC, and Lai E (2004) A new cluster validity measure and its application to image compression, Pattern Analysis and Applications 7(2), 205-220.

Clark MC, Hall LO, Goldgof DB, Clarke LP, Velthuizen RP and Silbiger MS (1994) MRI segmentation using fuzzy clustering techniques. IEEE Eng Med Biol, 13, pp.730742.

Clerc M and Kennedy J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space, In IEEE Transactions on Evolutionary Computation (2002) 6(1), pp. 58-73.

Couzin ID, Krause J, James R, Ruxton GD, Franks NR (2002) Collective Memory and Spatial Sorting in Animal Groups, Journal of Theoretical Biology, 218, pp. 1-11

Cui X and Potok TE (2005) Document Clustering Analysis Based on Hybrid PSO+Kmeans Algorithm, Journal of Computer Sciences (Special Issue), ISSN 1549-3636, pp. 27-33.

Das S, Abraham A, and Konar A (2008) Automatic Kernel Clustering with Multi-Elitist Particle Swarm Optimization Algorithm, Pattern Recognition Letters, Elsevier Science, Volume 29, pp. 688-699.

Davies DL and Bouldin DW (1979) A cluster separation measure, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1, 224227.

Deb K, Pratap A, Agarwal S, and Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. on Evolutionary Computation, Vol.6, No.2, April 2002.

Deneubourg JL, Goss S, Franks N, Sendova-Franks A, Detrain C and Chetien L (1991) The dynamics of collective sorting: Robot-like ants and ant-like robots. In Meyer JA and Wilson SW (Eds.) Proceedings of the First International Conference on Simulation of

Adaptive Behaviour: From Animals to Animats 1, pp. 356363. MIT Press, Cambridge, MA.

Dorigo M, Maniezzo V and Colorni A (1996), The ant system: Optimization by a colony of cooperating agents, IEEE Trans. Systems Man and Cybernetics Part B, vol. 26.

Dorigo M and Gambardella LM (1997) Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evolutionary Computing, vol. 1, pp. 5366.

Duda RO and Hart PE (1973) Pattern Classification and Scene Analysis. John Wiley and Sons, USA.

Dunn JC (1974) Well separated clusters and optimal fuzzy partitions. J. Cybern. 4, 95-104.

Eberhart RC and Shi Y (2001) Particle swarm optimization: Developments, applications and resources, In Proceedings of IEEE International Conference on Evolutionary Computation, vol. 1, pp. 81-86.

Evangelou IE, Hadjimitsis DG, Lazakidou AA, Clayton C (2001) Data Mining and Knowledge Discovery in Complex Image Data using Artificial Neural Networks, Workshop on Complex Reasoning an Geographical Data, Cyprus.

Everitt BS (1993) Cluster Analysis. Halsted Press, Third Edition.

Falkenauer E (1998) Genetic Algorithms and Grouping Problems, John Wiley and Son, Chichester.

Forgy EW (1965) Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of classification, Biometrics, 21.

Frigui H and Krishnapuram R (1999) A Robust Competitive Clustering Algorithm with Applications in Computer Vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5), pp. 450-465.

Gath I and Geva A (1989) Unsupervised optimal fuzzy clustering. IEEE Transactions on PAMI, 11, pp. 773-781.

Girolami M (2002) Mercer kernel-based clustering in feature space. IEEE Trans. Neural Networks 13(3), 780784.

Goldberg DE (1975) Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.

Grosan C, Abraham A and Monica C (2006) Swarm Intelligence in Data Mining, in Swarm Intelligence in Data Mining, Abraham A, Grosan C and Ramos V (Eds), Springer, pp. 1-16.

Hall LO, zyurt IB and Bezdek JC (1999) Clustering with a genetically optimized approach, IEEE Trans. Evolutionary Computing 3 (2) pp. 103112.

Handl J, Knowles J and Dorigo M (2003) Ant-based clustering: a comparative study of its relative performance with respect to k-means, average link and 1D-som. Technical Report TR/IRIDIA/2003-24. IRIDIA, Universite Libre de Bruxelles, Belgium

Handl J and Meyer B (2002) Improved ant-based clustering and sorting in a document retrieval interface. In Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII), volume 2439 of LNCS, pp. 913923. Springer-Verlag, Berlin, Germany.

Hertz T, Bar A, and Daphna Weinshall, H (2006) Learning a Kernel Function for Classification with Small Training Samples, Appearing in Proceedings of the 23rd International Conference on Machine Learning, Pittsburgh, PA.

Hoe K, Lai W, and Tai T (2002) Homogenous ants for web document similarity modeling and categorization. In Proceedings of the Third International Workshop on Ant Algorithms (ANTS 2002), volume 2463 of LNCS, pp. 256261. Springer-Verlag, Berlin, Germany.

Holland JH (1975) Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

Huang Z and Ng MG (1999) A fuzzy k-modes algorithm for clustering categorical data. IEEE Trans. Fuzzy Systems 7 (4), 446452.

Jain AK, Murty MN and Flynn PJ (1999) Data clustering: a review, ACM Computing Surveys, vol. 31, no.3, pp. 264323.

Kanade PM and Hall LO (2003) Fuzzy Ants as a Clustering Concept. In Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS03), pp. 227-232.

Kaufman, L and Rousseeuw, PJ (1990) Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons, New York.

Kennedy J, Eberhart R and Shi Y (2001) Swarm Intelligence, Morgan Kaufmann Academic Press.

Kennedy J and Eberhart R (1995) Particle swarm optimization, In Proceedings of IEEE International conference on Neural Networks, pp. 1942-1948.

Kim D W, Lee KY, Lee D, Lee KH (2005) A kernel-based subtractive clustering method. Pattern Recognition Letters 26(7), 879-891.

Kohonen T (1995) Self-Organizing Maps, Springer Series in Information Sciences, Vol 30, Springer-Verlag.

Konar A (2005) Computational Intelligence: Principles, Techniques and Applications, Springer.

Krause J and Ruxton GD (2002) Living in Groups. Oxford: Oxford University Press.

Kuntz P, Snyers D and Layzell P (1998) A stochastic heuristic for visualising graph clusters in a bi-dimensional space prior to partitioning. Journal of Heuristics, 5(3), pp. 327351.

Kuntz P and Snyers D (1994) Emergent colonization and graph partitioning. In Proceedings of the Third International Conference on Simulation of Adaptive Behaviour: From Animals to Animats 3, pp. 494 500. MIT Press, Cambridge, MA.

Kuntz P and Snyers D (1999) New results on an ant-based heuristic for highlighting the organization of large graphs. In Proceedings of the 1999 Congress on Evolutionary Computation, pp. 14511458. IEEE Press, Piscataway, NJ.

Leung Y, Zhang J and Xu Z (2000) Clustering by Space-Space Filtering, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (12), pp. 1396-1410.

Lewin B (1995) Genes VII. Oxford University Press, New York, NY.

Lillesand T and Keifer R (1994) Remote Sensing and Image Interpretation, John Wiley & Sons, USA.

Lumer E and Faieta B (1994) Diversity and Adaptation in Populations of Clustering Ants. In Proceedings Third International Conference on Simulation of Adaptive Behavior: from animals to animates 3, Cambridge, Massachusetts MIT press, pp. 499-508.

Lumer E and Faieta B (1995) Exploratory database analysis via self-organization, Unpublished manuscript.

MacQueen J (1967) Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, pp. 281-297.

Major PF, Dill LM (1978) The three-dimensional structure of airborne bird flocks. Behavioral Ecology and Sociobiology, 4, pp. 111-122.

Mao J and Jain AK (1995) Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks. vol. 6, 296317.

Milonas MM (1994) Swarms, phase transitions, and collective intelligence, In Langton CG Ed., Artificial Life III, Addison Wesley, Reading, MA.

Mitchell T (1997) Machine Learning. McGraw-Hill, Inc., New York, NY.

Mitra S, Pal SK and Mitra P (2002) Data mining in soft computing framework: A survey, IEEE Transactions on Neural Networks, Vol. 13, pp. 3-14.

Monmarche N, Slimane M and Venturini G (1999) Ant Class: discovery of clusters in numeric data by a hybridization of an ant colony with the k means algorithm. Internal Report No. 213, E3i, Laboratoire dInformatique, Universite de Tours

Moskovitch R, Elovici Y, Rokach L (2008) Detection of unknown computer worms based on behavioral classification of the host, Computational Statistics and Data Analysis, 52(9):4544–4566.

Ng R and Han J (1994) Efficient and effective clustering method for spatial data mining. In: Proc. 1994 International Conf. Very Large Data Bases (VLDB94). Santiago, Chile, September pp. 144155.

Omran M, Salman A and Engelbrecht AP (2002) Image Classification using Particle Swarm Optimization. In Conference on Simulated Evolution and Learning, volume 1, pp. 370374.

Omran M, Engelbrecht AP and Salman A (2005) Particle Swarm Optimization Method for Image Clustering. International Journal of Pattern Recognition and Artificial Intelligence, 19(3), pp. 297322.

Omran M, Salman A and Engelbrecht AP (2005) Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification. Fifth World Enformatika Conference (ICCI 2005), Prague, Czech Republic.

Pakhira MK, Bandyopadhyay S, and Maulik U (2004) Validity index for crisp and fuzzy clusters, Pattern Recognition Letters, 37, 487501.

Pal NR, Bezdek JC and Tsao ECK (1993) Generalized clustering networks and Kohonens self-organizing scheme. IEEE Trans. Neural Networks, vol 4, 549557.

Partridge BL, Pitcher TJ (1980) The sensory basis of fish schools: relative role of lateral line and vision. Journal of Comparative Physiology, 135, pp. 315-325.

Partridge BL (1982) The structure and function of fish schools. Science American, 245, pp. 90-99.

Paterlini S and Krink T (2006) Differential Evolution and Particle Swarm Optimization in Partitional Clustering. Computational Statistics and Data Analysis, vol. 50, pp. 1220 1247.

Paterlini S and Minerva T (2003) Evolutionary Approaches for Cluster Analysis. In Bonarini A, Masulli F and Pasi G (eds.) Soft Computing Applications. Springer-Verlag, Berlin. 167-178.

Pirooznia M and Deng Y: SVM Classifier a comprehensive java interface for support vector machine classification of microarray data, in Proc of Symposium of Computations in Bioinformatics and Bioscience (SCBB06), Hangzhou, China.

Ramos V, Muge F and Pina P (2002) Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies. Soft Computing Systems: Design, Management and Applications. 87, pp. 500509.

Ramos V and Merelo JJ (2002) Self-organized stigmergic document maps: Environments as a mechanism for context learning. In Proceedings of the First Spanish Conference on Evolutionary and Bio-Inspired Algorithms (AEB 2002), pp. 284293. Centro Univ. Merida, Merida, Spain.

Rao MR (1971) Cluster Analysis and Mathematical Programming,. Journal of the American Statistical Association, Vol. 22, pp 622-626.

Ratnaweera A and Halgamuge KS (2004) Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, In IEEE Trans. on Evolutionary Com-

putation 8(3): 240-254.
Rokach, L., Maimon, O.: (2005), Clustering Methods, Data Mining and Knowledge Discovery Handbook, Springer, pp. 321-352.
Rosenberger C and Chehdi K (2000) Unsupervised clustering method with optimal estimation of the number of clusters: Application to image segmentation, in Proc. IEEE International Conference on Pattern Recognition (ICPR), vol. 1, Barcelona, pp. 1656-1659.
Sarkar M, Yegnanarayana B and Khemani D (1997) A clustering algorithm using an evolutionary programming-based approach, Pattern Recognition Letters, 18, pp. 975986.
Scholkopf B and Smola AJ (2002) Learning with Kernels. The MIT Press, Cambridge.
Selim SZ and Alsultan K (1991) A simulated annealing algorithm for the clustering problem. Pattern recognition, 24(10), pp. 1003-1008.
Shi Y and Eberhart RCD (1999) Empirical Study of particle swarm optimization, In Proceedings of IEEE International Conference Evolutionary Computation, Vol. 3, 101-106.
Storn R and Price K (1997) Differential evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 11(4), pp. 341359.
Tsang W and Kwong S (2006) Ant Colony Clustering and Feature Extraction for Anomaly Intrusion Detection, in Swarm Intelligence in Data Mining, Abraham A, Grosan C and Ramos V (Eds), Springer, pp. 101-121.
Vapnik VN (1998) Statistical Learning Theory. Wiley, New York.
Wang X, Wang Y and Wang L (2004) Improving fuzzy c-means clustering based on feature-weight learning. Pattern Recognition Letters, vol. 25, pp. 112332.
Xiao X, Dow ER, Eberhart RC, Miled ZB and Oppelt RJ (2003) Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization, Proc of the 17th International Symposium on Parallel and Distributed Processing (PDPS '03), IEEE Computer Society, Washington DC.
Xu, R., Wunsch, D.: (2005), Survey of Clustering Algorithms, IEEE Transactions on Neural Networks, Vol. 16(3): 645-678
Xu R and Wunsch D (2008) Clustering, IEEE Press Series on Computational Intelligence, USA.
Zahn CT (1971) Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Transactions on Computers C-20, 6886.
Zhang T, Ramakrishnan R and Livny M (1997) BIRCH: A New Data Clustering Algorithm and Its Applications, Data Mining and Knowledge Discovery, vol. 1, no. 2, pp. 141-182.
Zhang DQ and Chen SC (2003) Clustering incomplete data using kernel-based fuzzy c-means algorithm. Neural Process Letters 18, 155162.
Zhang R and Rudnicky AI (2002) A large scale clustering scheme for kernel k-means. In: The Sixteenth International Conference on Pattern Recognition, p. 289292.
van den Bergh F and Engelbrecht AP (2001) Effects of swarm size on cooperative particle swarm optimizers, In Proceedings of GECCO-2001, San Francisco CA, 892-899.
van der Merwe DW and Engelbrecht AP (2003) Data clustering using particle swarm optimization. In: Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 215-220, Piscataway, NJ: IEEE Service Center

Index

C-medoids, 12

Fuzzy C-means, 12

K-means, 12

Pattern, 10